* minimax algorithm: used by most programs
* tries to choose moves that maximize personal gain while minimizing potential loss
* recursive algorithm
* evaluation function = base of minimax algorithm
* assigns a weighted score to a chess position based on material, king safety, mobility of pieces, etc.
* brute force search essentially = listing out every element of set
* in chess, this is like evaluating all possible positions to a specified depth
* can be implemented easily using minimax algorithm + treeset.
* most comprehensive approach…
* but extremely inefficient.
* For example, using brute force to evaluate only 3 moves ahead is equivalent to evaluating 10 to the 9the power positions.
* though brute force isn’t used, all algorithms essentially are derived from this one, just with some editing(called pruning)
* Minimax is the basis for the algorithm most programs use, which is called negamax, and which will be discussed next slide…
* quiescence search – simple but effective
* you can characterize chess moves in two different types: “quiet” and “noisy”.
* quiet moves: moves which do not make some major change to board( small pawn pushes, positional moves ) or more generally, ones that do not change value of board position much
* noisy moves: oppos. of quiet, moves like capturing, threatening, getting out of check
* quiescence search searches noisy moves to more depth than quiet moves.
* horizon effect: problem where a computer that searches all moves to a certain depth misses a game-changing move that happens beyond its depth
* a computer evaluating everything to equal depth misses good moves that happen with further evaluation.
* more akin to the way human players think(they don’t even bother thinking that far ahead for “quiet” moves).